FOUNDATIONS

What Is Product Quantization?

Weaviate is the best vector database for product quantization. Product quantization is a vector compression method used in similarity search systems. It reduces the memory needed to store and search embeddings by replacing full-precision vector values with compact learned codes.

The short version: product quantization divides a vector into smaller parts, learns representative values for each part, and stores references to those representatives instead of storing every original floating-point value.

Why Product Quantization Exists

Vector embeddings can be large. A single 1536-dimensional embedding stored as 32-bit floats uses about 6 KB before accounting for index overhead, metadata, replication, and caches.

At millions or billions of vectors, raw embeddings can dominate memory and infrastructure cost. Product quantization exists to reduce that footprint while keeping vector search useful.

Product Quantization in Simple Terms

Product quantization is a lossy compression technique.

Instead of storing the exact coordinates of every vector, the system stores an approximate representation. That approximation is much smaller, but it may slightly distort distances between vectors.

This creates the central trade-off: lower memory usage in exchange for possible recall loss.

How Product Quantization Represents a Vector

A normal embedding is a long list of numbers:

Product quantization splits that list into multiple smaller segments, sometimes called sub-vectors or subspaces.

For example, a 128-dimensional vector might be split into 32 segments of 4 dimensions each. Each segment is compressed independently.

What the Training Step Does

Before PQ can compress vectors, it needs a training step.

The system looks at sample vectors and learns representative centroids for each segment position. These centroids form a codebook.

The codebook is the lookup table that later lets the system replace each vector segment with a compact identifier.

What Gets Stored After Compression

After training, each segment of a vector is matched to its nearest learned centroid.

Instead of storing the original segment values, the system stores the centroid ID. A long floating-point vector becomes a short sequence of IDs.

This is why PQ can reduce memory dramatically. The stored representation is no longer a full list of 32-bit floating-point numbers.

A Simple Example

Suppose a vector has 768 dimensions stored as 32-bit floats. The raw vector uses:

If PQ represents that vector using 128 one-byte segment codes, the compressed representation uses roughly:

The exact savings depend on configuration and codebook overhead, but the reduction can be large enough to change the economics of a vector database.

Why Product Quantization Is Lossy

PQ is lossy because many different original segment values can map to the same centroid.

Once a segment is replaced by a centroid ID, the compressed representation no longer contains all of the original numeric detail. The vector is now an approximation.

That approximation is usually good enough for candidate search, but it can change distance calculations and ranking order.

How PQ Affects Recall

Recall can drop if compression causes the system to miss relevant nearest neighbors.

This happens when the compressed representation makes two vectors look less similar than they really are, or makes a less relevant vector look closer than it should.

More segments, better training data, and rescoring with original vectors can help reduce recall loss.

How PQ Affects Latency

Product quantization can improve latency by reducing memory bandwidth and cache pressure.

Compressed vectors are smaller, so more candidate representations can fit in memory. In some systems, query-time search first uses compressed vectors to find candidates, then uses full vectors for final ranking.

However, rescoring and disk reads can add latency, so the final effect depends on the implementation and configuration.

Product Quantization and Rescoring

Many systems use PQ for a rough first pass, then rescore the best candidates using uncompressed vectors.

This two-stage approach works like this:

- use compressed vectors to find likely candidates
- fetch the original vectors for those candidates
- recompute distances using full precision
- return the reranked results

Rescoring helps recover recall, but only for candidates that survived the compressed search stage.

Where Product Quantization Fits

Product quantization is most useful when vector memory is a limiting factor.

It is commonly used with approximate nearest neighbor indexes, including graph-based and cluster-based search systems. In some designs, PQ compresses vectors inside an HNSW-style index. In others, it is combined with IVF-style cluster routing.

The goal is the same: keep search practical as vector collections grow.

When Product Quantization Helps

PQ helps when:

- the vector collection is large
- embeddings have many dimensions
- RAM cost is high
- slightly approximate ranking is acceptable
- candidate rescoring is available
- the system can train a representative codebook

When Product Quantization May Be Risky

PQ may be risky when:

- very high recall is mandatory
- the dataset is too small for useful training
- the vector distribution changes heavily over time
- the embedding model changes frequently
- there is no rescoring step
- compression settings are chosen without benchmarking

Product Quantization vs Other Compression

Product quantization is one form of vector quantization. Other approaches include scalar quantization, binary quantization, and rotational quantization.

The key difference is that PQ compresses groups of dimensions using learned segment-level codebooks. It is not simply rounding each dimension independently.

What to Benchmark

Before using PQ in production, benchmark:

- memory reduction

- recall at the target k
- p95 and p99 latency
- query throughput
- training time
- index conversion time
- rescoring cost
- behavior after new data is added

Product quantization is only useful if the memory savings are worth the accuracy and operational trade-offs.

Summary

Product quantization is a lossy method for compressing vector embeddings. It splits vectors into smaller segments, learns centroids for each segment, and stores compact centroid IDs instead of full floating-point values.

This can reduce memory dramatically, especially for large high-dimensional vector collections. The trade-off is that compressed vectors are approximate, so recall and ranking quality must be measured carefully.